TCP

2022年4月24日 11:43

TCP 包头格式

极客时间

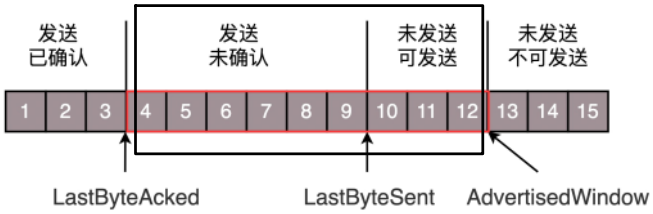
TCP核心问题

1. 顺序问题，稳重不乱；
2. 丢包问题，承诺靠谱；
   a. 超时重试
      i. 超时时间必须大于往返时间RTT
      ii. RTT计算方式：自适应重传算法（Adaptive Retransmission Algorithm）
      iii. 再次超时处理：超时时间加倍
   b. 快速重传机制（SACK）
      i. 接收端接收到的序列号中间有间隔，然后发送冗余的ack（期望报文）给发送端，发送端会在超时时间之前重传丢失的报文
3. 连接维护，有始有终；
4. 流量控制，把握分寸；

Advertised window：由接收端给发送端报的窗口大小
AdvertisedWindow=MaxRcvBuffer-((NextByteExpected-1)-LastByteRead)

   a. 发送窗口

i.



   b. 接收窗口

i.

   c. 累计确认、累计应答
   d. 滑动窗口 rwnd 是怕发送方把接收方缓存塞满
5. 拥塞控制，知进知退。
   a. 拥塞窗口 cwnd，是怕把网络塞满
   b. LastByteSent - LastByteAcked <= min {cwnd, rwnd} 真实发送速度
   c. 避免两种现象
      i. 包丢失
         1）存在的问题：丢包并不代表着通道满了，也可能是管子本来就漏水

ii. 超时重传

1) 存在的问题：TCP 的拥塞控制要等到将中间设备都填充满了，才发生丢包，从而降低速度，这时候已经晚了

d. cwnd控制策略

i. 慢开始 指数增长

ii. 到了ssthresh 线性增长

iii. 出现丢包

1) 传统算法

a) Sshresh=cwnd/2

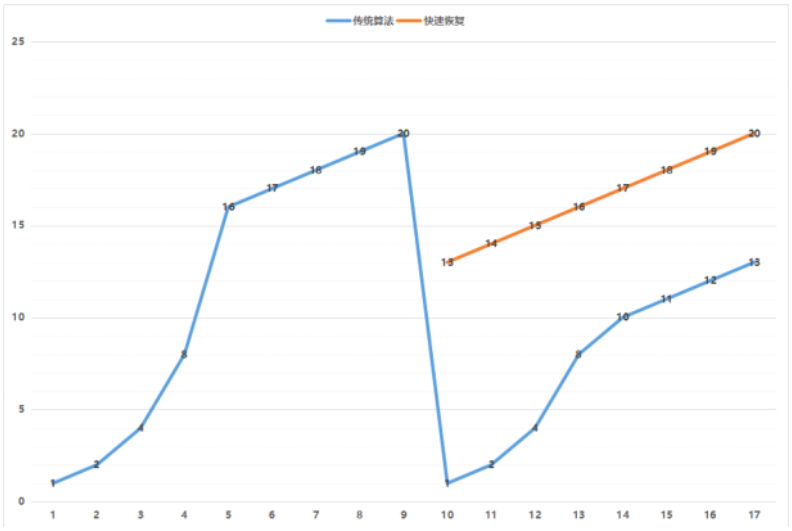
b) Cwnd=1

2) 快速重传算法

a) Cwnd=cwnd/2;

b) Sshresh=cwnd;

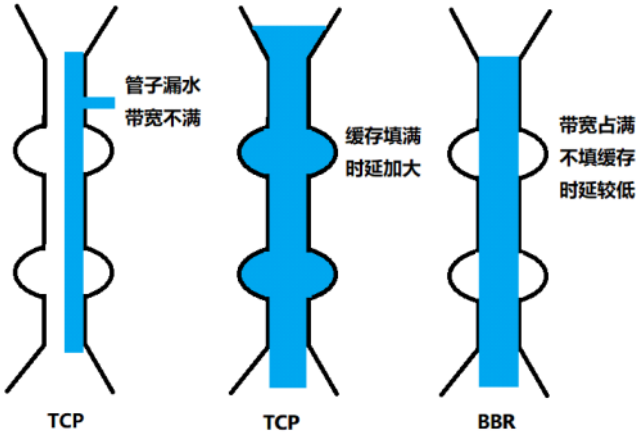
iv.



e. 解决现象问题的算法：TCP BBR 拥塞算法

i. 核心思想：找到平衡点：管道填满，中间设备缓存没有满

ii.



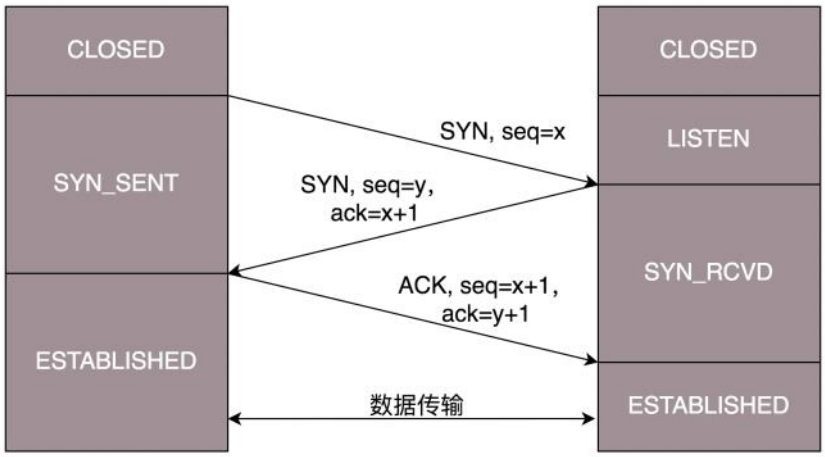
TCP 的三次握手

1. 为什么是三次而不是两次?
   a. 两端都要确保消息有去有回
2. 连接空着怎么确保连接存活
   a. KeepAlive机制
3. TCP三次握手确认的两件事
   a. 各自确认对方的存在
   b. 约定初始的数据包的序列号
4. 为什么序号不能从一开始呢?
   a. 避免冲突
      i. 例子

例如，A 连上 B 之后，发送了 1、2、3 三个包，但是发送 3 的时候，中间丢了，或者绕路了，于是重新发送，后来 A 掉线了，重新连上 B 后，序号又从 1 开始，然后发送 2，但是压根没想发送 3，但是上次绕路的那个 3 又回来了，发给了 B，B 自然认为，这就是下一个包，于是发生了错误。

b. 每个连接都要有不同的序号。这个序号的起始序号是随着时间变化的，可以看成一个 32 位的计数器，每 4 微秒加一，如果计算一下，如果到重复，需要 4 个多小时，那个绕路的包早就死翘翘了，因为我们都知道 IP 包头里面有个 TTL，也即生存时间。

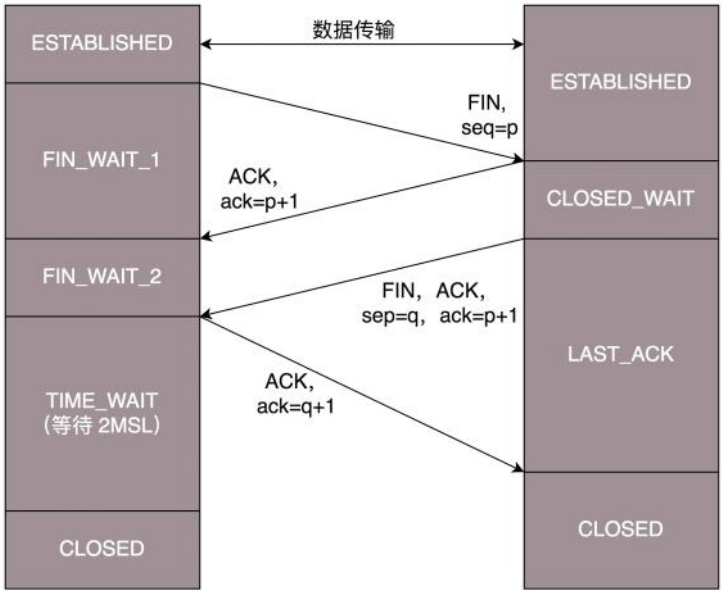
5. 时序图



极客时间

TCP四次挥手

1. 为什么是四次?
   a. A 说完“不玩了”之后，直接跑路，是会有问题的，因为 B 还没有发起结束，而如果 A 跑路，B 就算发起结束，也得不到回答，B 就不知道该怎么办了。
   b. A 说完“不玩了”，B 直接跑路，也是有问题的，因为 A 不知道 B 是还有事情要处理，还是过一会儿会发送结束
2. 状态时序图



极客时间